Visvesvaraya Technological University
Belagavi-590018 Karnataka

A Mini Project Report on

"Action Detection For Sign Language Digital Image Processing "

Mini Project Report Submitted in partial fulfillment of the requirements for the Digital Image Processing Lab

(18AIL68)

Bachelor of Engineering

In

Artificial Intelligence and Machine Learning

Submitted by

A S Sushmitha Urs	**1JT20AI001**
Vaibhavi B Raj	**1JT20AI047**

Under The Guidance of

Mrs. Soumya K N
Assistant Professor
Dept. of AIML
JIT, Bengaluru

Dr.Madhu B R
Professor and HOD
Dept. of AIML
JIT, Bengaluru

Jyothy Charitable Trust®
Jyothy Institute of Technology
Tataguni, off Kanakapura road, Bengaluru-560082
Approved by The **A**ll **I**ndia **C**ouncil for **T**echnical **E**ducation (**AICTE)** - New Delhi;
Affiliated to **V**isvesvaraya **T**echnological **U**niversity **(VTU)**, Belagavi;

Department of Artificial Intelligence and Machine Learning

CERTIFICATE

This is to certify that the project work titled **"ACTION DETECTION FOR USING DIGITAL IMAGE PROCESSING"** is carried out by **AS Sushmitha Urs[1JT20AI001], Vaibhavi B Raj [1JT20AI047],**a Bonafede students of Bachelor of Technology at the Jyothy Institute of Technology, Bangalore in partial fulfillment for the award of degree in Bachelor of Technology in Artificial Intelligence and Machine Learning, during the year 2023-2024.

Prof. Soumya K N
Assistant Professor
Dept. of AIML,
JIT, VTU,
Bangalore
Date:

Dr. Madhu B R
Professor
Dept. of AIML,
JIT, VTU,
Bangalore
Date:

Name of the Examiner

Signature of Examiner

1.

2.

DECLARATION

We, **A S Sushmitha Urs(1JT20AI001), Vaibhavi B Raj(1JT20AI047),** are students of sixth semester B.Tech in **Artificial Intelligence and Machine Learning** at Jyothy Institute of Technology, **VTU**, hereby declare that the project titled **"Action Detection For Sign Language"** has been carried out by us and submitted in partial fulfilment for the award of degree in **Bachelor of Technology in Artificial Intelligence and Machine Learning** during the academic year **2023-2024**. Further, the matter presented in the project has not been submitted previously by anybody for the award of any degree or any diploma to any other University, to the best of our knowledge and faith.

Signature

Name1: **A S Sushmitha Urs**
USN: **1JT20AI001**
Name2: **Vaibhavi B Raj**
USN: **1JT20AI047**

PLACE : BANGALORE
DATE :

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of a large number of individuals who have been responsible for the successful completion of this project work.

First, we take this opportunity to express our sincere gratitude to ***"Jyothy Institute of Technology"****, VTU for providing us with a great opportunity to pursue our Bachelor's Degree in this institution.*

In particular we would like to thank ***Dr. Gopalkrishnan****, Principal, Jyothy Institute of Technology, VTU for their constant encouragement and expert advice.*

It is a matter of immense pleasure to express our sincere thanks to ***Dr. Madhu B R, Head of the department****,* ***Artificial Intelligence and Machine Learning****, VTU****,*** *for providing right academic guidance that made our task possible.*

We would like to thank our guide ***Prof. Soumya K N*** *Assistant Professor,* ***Dept. of Artificial Intelligence and Machine Learning****, for sparing his/her valuable time to extend help in every step of our project work, which paved the way for smooth progress and fruitful culmination of the project.*

We would like to thank our Project Coordinator ***Prof. Soumya K N*** *and all the staff members AIML for their support.*

We are also grateful to our family and friends who provided us with every requirement throughout the course.

We would like to thank one and all who directly or indirectly helped us in completing the Project work successfully.

Signature of Students

ABSTRACT

This project aims to improve communication accessibility for individuals with hearing impairment disabilities through the development of an efficient and fast algorithm for identifying the alphabets in American Sign Language (ASL) using natural hand gestures. The system's ultimate objective is to function as a translator between sign language and spoken language, allowing for more effective and efficient communication between hearing-impaired and normal individuals. The project utilizes image processing, machine learning, and artificial intelligence using CNN to identify ASL gestures and produce easily understandable outputs. This work has the potential to significantly enhance communication inclusivity for individuals with hearing disabilities.

TABLE OF CONTENTS

LIST OF FIGURES

NOMENCLATURE USED

CNN	Convolutional Neural Network
ASL	American Sign Language
HCI	human-computer interaction
LSTM	Long short-term memory
DCNN	Deep convolutional neural networks
SVM	Support Vector Machine
ReLu	rectified linear activation unit
MLP	Multi-Layer Perceptron

Chapter 1:

1.INTRODUCTION

Sign language alphabets serve as a vital tool in facilitating communication between individuals with hearing and speech impairments and those without disabilities. With the aid of technology, sign language gestures can now be accurately translated into readable text, greatly reducing communication barriers. The use of sign languages extends beyond national borders, with over 100 different sign languages being used globally. Examples include Indian Sign Language, American Sign Language (ASL), and Italian Sign Language. ASL stands out as the most widely used sign language worldwide, with over a million speakers in the United States and more than 30 other countries. ASL is a highly intricate language that employs a combination of hand, finger, and facial gestures to convey the thoughts and ideas of individuals with speech impairments. Compared to spoken languages like Italian or German, ASL exhibits a greater degree of variability, making it a linguistically rich and nuanced form of communication. Through ASL, individuals with speech impairments find hope and happiness, and its positive impact reaches far and wide. The American Manual Alphabet comprises 26 distinct gesture signs used in ASL to represent various English words. Within ASL, 19 different hand shapes are utilized, which collectively form the 26 alphabets of the American manual alphabet. It is worth noting that certain hand shapes convey different letters depending on their orientation. While various approaches to sign language recognition have been proposed, this particular paper focuses on utilizing monocular camera images as the sole input source for the recognition task. To achieve this, the study incorporates important libraries and modules such as TensorFlow, Matplotlib, OpenCV, and CVZone. These powerful tools enable the implementation of convolutional neural networks (CNNs) as the chosen classification technique. The paper will explore the intricacies of data acquisition, the specific CNN topology employed, and the results obtained from the research endeavour. The potential of developing real-time sign language recognition systems based on these findings is significant, as it would greatly enhance communication accessibility for individuals who are not proficient in ASL. ASL represents not only a means of communication but also a remarkable form of interactive art. Its impact resonates deeply within a wide community of individuals with speech and hearing disabilities, fostering inclusivity and empowerment. The utilization of TensorFlow, OpenCV, Matplotlib, and CVZone libraries empowers researchers

and developers to leverage cutting-edge technologies in the field of sign language recognition. These libraries provide robust tools for image processing, computer vision, and deep learning, enabling accurate and efficient recognition of sign language gestures. The study aims to contribute to the advancement of sign language recognition systems by utilizing state-of-the-art technologies and methodologies. By leveraging the power of CNNs and the information extracted from monocular camera images, the research holds the potential to revolutionize real-time sign language translation and communication assistance .In addition to its practical implications, the research also highlights the importance of recognizing ASL as a legitimate language with characteristics akin to spoken languages. The linguistic complexity and expressive nature of ASL make it a rich and vibrant means of communication, deserving of further exploration and understanding. The development of efficient sign language recognition systems can bring about positive social change by fostering greater inclusivity and improving communication accessibility for individuals with speech and hearing impairments. It has the potential to enhance education, employment opportunities, and social interactions for this community. By combining the power of technology with the beauty of sign language, the research aims to bridge the communication gap and create a more inclusive society. Through the recognition and understanding of sign language, barriers are broken down, and individuals with speech and hearing disabilities are empowered to express themselves fully. The field of sign language recognition is a dynamic and rapidly evolving area of research. Continued advancements in technology, such as the integration of deep learning models, image processing techniques, and real-time systems, hold great promise for the future development of more sophisticated sign language recognition systems.

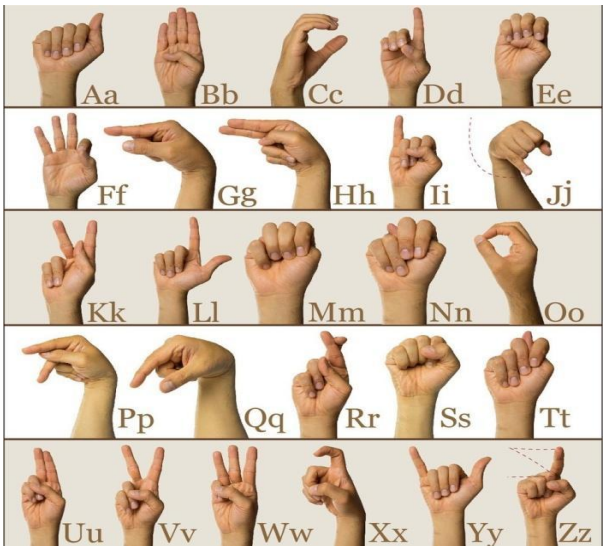


Fig 1.1 ASL Poses

Chapter 2:
2.Literature Survey

The system captures input images from live videos using a low-cost device like a webcam and preprocesses the hand gesture images through various steps, including color space conversion, binarization, erosion, and hole filling. The model used is CNN. The fused features are then used for gesture classification through a softmax classifier, achieving high recognition accuracy for fifteen common words in real-time compared to state-of-the-art systems.. As a result, the average acceptance of gesture-based sign word recognition is 96.96% which leads to better results than state-of-art system.[1]

The proposed model utilizes video sequences and extracts both temporal and spatial features from them. Inception, a Convolution Neural Network (CNN), is employed to recognize spatial features, while a Recurrent Neural Network (RNN) is used to train on temporal features.In the study, the model experienced a decrease in accuracy when faces were included in the videos. This occurred because the model learned incorrect features due to the variation in signers' faces. To mitigate this, the videos were trimmed to focus solely on gestures, limited to the area extending up to the neck. Additionally, the model struggled with clothing variations.The proposed CNN model achieved an accuracy of 93.05% .[2]

This paper presents a real-time hand gesture recognition system for speech-impaired individuals using Microsoft Kinect RGB-D camera data. Computer vision techniques enable the establishment of a one-to-one mapping between depth and RGB camera pixels, followed by hand gesture segmentation. Convolutional Neural Networks (CNNs) are trained on 36 static ISL gestures, achieving 98.81% accuracy, while Convolutional LSTMs achieve 99.08% accuracy on 10 ISL dynamic word gestures. The model demonstrates accurate real-time performance and shows promise for further research in sentence formation through gestures, as well as adaptability to ASL gestures. [3]

The Convolutional Neural Network (CNN) algorithm in Deep Learning is employed as a classification tool, leveraging its ability to learn and recognize various patterns.The study explores the effectiveness of combining CNN models using the Ensemble method, resulting in a significant increase in accuracy value to 99.4%. This demonstrates that the Ensemble approach enhances the performance of the system in accurately translating sign language gestures. [4].

The paper proposes a machine learning approach to classify single-hand alphabet letters in Czech Sign Language. The paper highlights the importance of sign language recognition systems in improving communication with the deaf community and discusses the challenges of recognizing sign language gestures due to their dynamic and non-linear nature.The proposed approach uses a Convolutional Neural Network (CNN) model.The proposed CNN model achieved an accuracy of 99.7% in recognizing single-hand alphabet letters in Czech Sign Language. [5].

This human-computer interaction (HCI) system enables communication between speech-impaired individuals and those unfamiliar with sign language, offering a solution to bridge the communication gap and reduce isolation. The paper presents a real-time American Sign Language (ASL) translation system that utilizes a Convolutional Neural Network (CNN) algorithm. The model consists of 8 layers, and it achieves an impressive accuracy rate of 98%. [6].

The proposed approach utilizes a DCNN model comprising convolutional layers, ReLU activation, max-pooling layers, batch normalization layers, and fully connected layers.The DCNN model consists of multiple convolutional layers with rectified linear unit (ReLU) activation, followed by max-pooling layers, batch normalization layers, and fully connected layers. The training data consists of images of hand gestures representing different American Sign Language alphabet letters and numbers. The data is preprocessed by resizing the images to a fixed size, converting them to grayscale, and applying normalization.The proposed DCNN model achieves an impressive accuracy of 99.66% in recognizing static hand gestures in ASL. [7].

The paper presents a real-time hand gesture recognition system for American Sign Language (ASL) using different algorithms. The system employs four algorithms: K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Convolutional Neural Network (CNN), and Multi-Layer Perceptron (MLP).The study utilizes a custom dataset consisting of 1200 images of 24 different ASL gestures for training and evaluation. The reported accuracy values for the algorithms are: KNN - 90.2%, SVM - 91.3%, CNN - 96.4%, and MLP - 93.7%.[8]

The strategies and algorithms for hand gesture recognition in sign language are presented in these papers. They strive to accurately translate sign language gestures so that people with speech impairments can converse with others. They use a range of models, including CNN, RNN, Convolution LSTMs, and DCNN, to extract temporal and spatial data from movies or images. With high accuracy ranging from 93.05% to 99.7%, the models accurately identify hand gestures for phrases, letters, and numbers in spoken languages including ASL and Czech Sign Language. The papers discuss the difficulties posed by changes in face and clothes as well as the necessity of processing methods for enhancing precision in recognition. The papers also demonstrate the advantages of the ensemble method for higher performance and real-time sign language communication, as well as the usage of customized datasets.

Chapter 3

3.Objective and Methodology

3.1 Objective

The basic objective of this project is to:

- Develop a computer based intelligent system that will enable hearing impairment people to communicate with other people using their natural hand gestures.
- To validate the usability of the system in real-world applications such as sign language translation, human-computer interaction, and virtual reality.

3.2 Methodology

1. Data Collection:

The code connects to a video source—in this case, the webcam—by using OpenCV's VideoCapture() function. The webcam capture is represented by the cap variable, which starts out as a VideoCapture object. The while loop's code uses the cap.read() function to continually read camera frames. This function returns two values: img, which contains the information from the captured frame, and success, which indicates if the frame was successfully read. Use the success variable to ascertain whether the frame was successfully read. If it returns true, it signifies a frame was properly captured. The img variable contains the frame data that corresponds to it. The code gathers a stream of visual data from the webcam by continually taking frames within the loop. The loop iterates representing a single frame capture, and it runsendlessly until manually broken. For various reasons, such as computer vision tasks, object detection, or creating datasets for machine learning applications, the gathered frames can be further processed, analyzed, or stored. Around 2000 images are collected as input Dataset.

2. Data Processing:

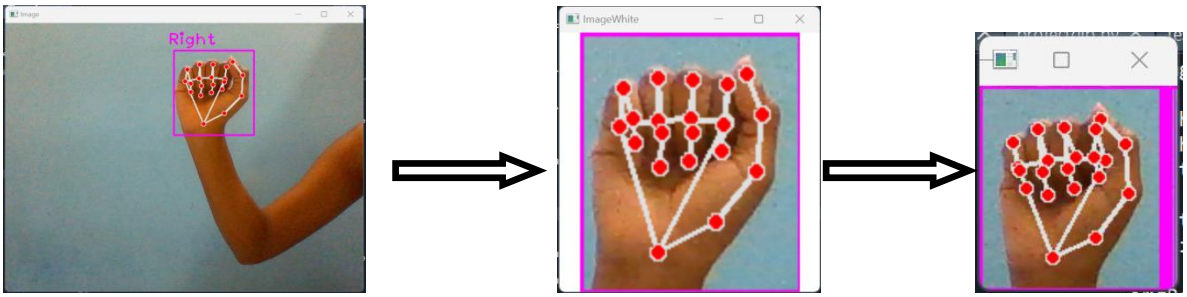


Fig 3.2.1 Data Processing

The programme processes the collected frames using computer vision techniques. Each frame's hand is recognized using the HandDetector module from the cvzone library. When a hand is found, the method resizes and crops the image to isolate and normalize the hand region. The edited hand image is shown in two windows:"ImageWhite" displays the edited hand image with a white background, and "ImageCrop" displays the edited hand region cropped. This tampering with the recorded frames might be thought of as data processing.

The flow chart for the following stages explains the data processing is shown in Figure 3.2:

1. Hand detection: The HandDetector class from the cvzone is used in the code. Using the HandTrackingModule module, the camera can recognize hands in video frames.

2. Image resizing and cropping The method crops the image based on the bounding box of a hand that was recognized and resizes it to a specified size. To accomplish this, the image's data must be changed. For instance, a portion of interest can be extracted and resized using OpenCV methods.

3. picture display: To display the collected video frame, the cropped hand picture (imgCrop), and the resized hand image (imgWhite), the code makes use of OpenCV's imshow() function. On the other side, the next section of the code is where data collecting is done.

4. Image saving: The code increases a counter and saves the current 'imgWhite' image to a specified folder with a unique filename based on the current timestamp when a user presses the "s" key. The code may thus record and save images of the hand motions or places.

3. **CNN Model:**

A family of deep learning models known as convolutional neural networks (CNNs) is frequently employed for image identification applications. They were created with the express purpose of capturing spatial relationships and patterns in photographs. Among the many different layers that make up CNNs are convolutional, pooling, and fully connected layers.

In our method, Teachable Machine is utilized to educate the model, which, as illustrated in Fig. 3.3, has a deep neural network structure made up of 28 CNN layers. Teachable Machine selects the appropriate architecture and hyper parameters according on the data provided throughout the training phase. Users can focus on the data and intended outcomes while the platform takes care of other things.

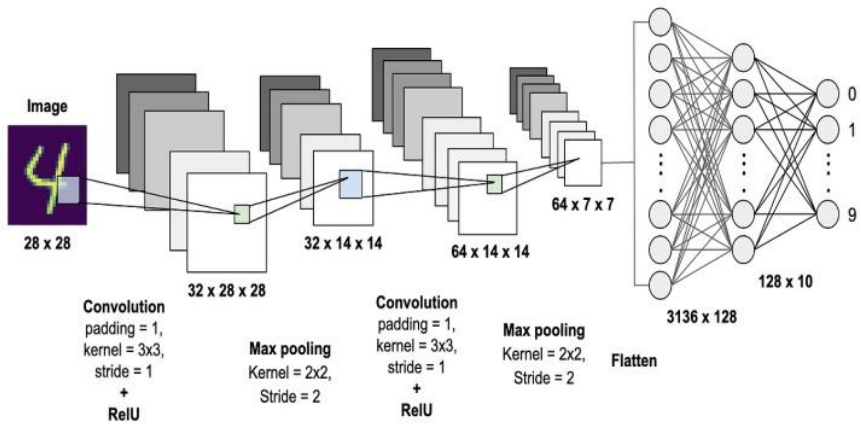


Fig 3.3.1 CNN Model

The first layer of the CNN to accept an input picture or feature map is called the input layer.

Convolutional Layers: By applying filters or kernels to the input, these layers carry out convolution processes. From the input, each convolutional layer pulls features that identify local relationships and spatial patterns.

Layers of Activation: The output of each convolutional layer is subjected to element-by-element application of activation functions, such as ReLU (Rectified Linear Unit). They give the network non-linearity, allowing it to pick up more intricate representations.

Pooling Layers: Pooling layers reduce the computational complexity and extract the most pertinent characteristics by downsampling the spatial dimensions of the feature maps. Max pooling and average pooling are two common pooling methods

Batch normalization layers: By normalizing the activations of the preceding layer, these layers increase the network's stability and training efficiency. layers: During training, dropout layers randomly change a portion of the input units to zero, which prevents overfitting by providing regularization and enhancing the network's generalization.

Fully connected layers: sometimes referred to as dense layers, link every neuron in the layer below to every neuron in the layer above. Fully connected layers extract high-level features and capture global dependencies.

The final CNN layer, known as the "output Layer," is responsible for creating the network's forecasts or outputs. The output layer's activation function, such as softmax for multi-class classification or sigmoid for binary classification, is determined by the task at hand

3. Data analysis:

The code incorporates the Classification Module from cvzone to perform data analysis. It uses a pre-trained classification model (keras_model.h5) and a corresponding labels file (labels.txt).The Classifier object is initialized with these files.The classifier receives the processed hand image (imgWhite) from the code. Obtain a prediction and its associated index using the getPrediction() function.This prediction represents the detected hand gesture or sign. The code's data processing makes use of the Classification Module from cvzone to enable gesture recognition using a pertained model and labels file. The pre-trained classification model (keras_model.h5) and the associated labels file (labels.txt) are initialized with a

Classifier object by the code. The Classifier's getPrediction() method receives the preprocessed hand image (imgWhite) and returns a gesture prediction along with the corresponding index.Based on the classification produced by the trained model, the prediction indicates the hand gesture or sign that was detected.

Real-time recognition and identification of hand motions from the collected frames is made possible by this technique. For further processing and analysis, the anticipated gesture label and its accompanying index are printed to the console.The Classification Module from cvzone is used in the code's data analysis to enable gesture recognition utilizing a pre-trained model and labels file. The pre-trained classification model (keras_model.h5) and the associated labels file (labels.txt) are initialized with a Classifier object by the code.

The Classifier's getPrediction() method receives the preprocessed hand image (imgWhite) and returns a gesture prediction along with the corresponding index.Based on the classification produced by the trained model, the prediction indicates the hand gesture or sign that was detected. Real-time recognition and identification of hand motions from the collected frames is made possible by this technique. For further processing and analysis, the anticipated gesture label and its accompanying index are printed to the console.

The results of the gesture recognition can be used for a range of tasks, such as sign language interpretation, gesture-based control, and interactive user interfaces. Accurate gesture analysis and identification depend on the training model and labels file. The code exemplifies the integration of the categorization module to enable real-time and efficient gesture processing. The analysis of the recognised gestures opens up potential for interactive applications and human-computer interaction and enables subsequent actions or decisions to be made in response to the identified hand motions.

4. Prediction:

Using the processed image as a basis, the code trains a classification model to predict a hand gesture or sign. Calling the getPrediction() function of the Classifier object returns the prediction. The processed hand image is analyzed by the getPrediction() method, which then applies the classification model to produce the prediction.

The label or class assigned to the discovered hand motion is represented by the prediction. The getPrediction() method also returns the index corresponding to the predicted class along with the predicted label. In order to facilitate additional research or the mapping of the anticipated class to particular actions, the index gives a numerical representation of the expected class.

The code gains more knowledge about the recognised hand gesture or sign as a result of getting the prediction and index. This output allows for the prediction of future actions or reactions based on the recognised gesture, as well as the comprehension and analysis of hand movements. Using the prediction and index, a virtual interface can be controlled, specific functionality may be started, and feedback based on the detected gesture can be supplied. Because of the trained classification model's accuracy, gesture analysis as a whole is improved, and interactive and natural human-computer interaction is made possible.

Chapter 4

4. System Design

4.1 System Architecture

The system is a vision based approach. All the signs are represented with

Bare hands and so it eliminates the problem of using any artificial devices for interaction.

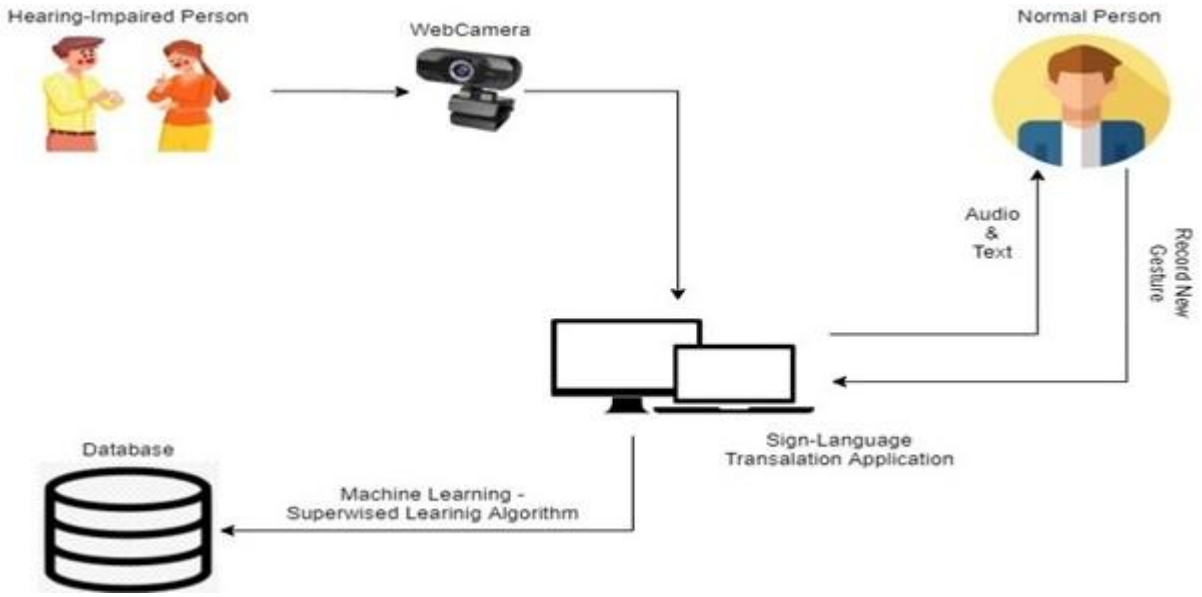


Fig 4.1.1 System Architecture

4.2 Use Case diagram

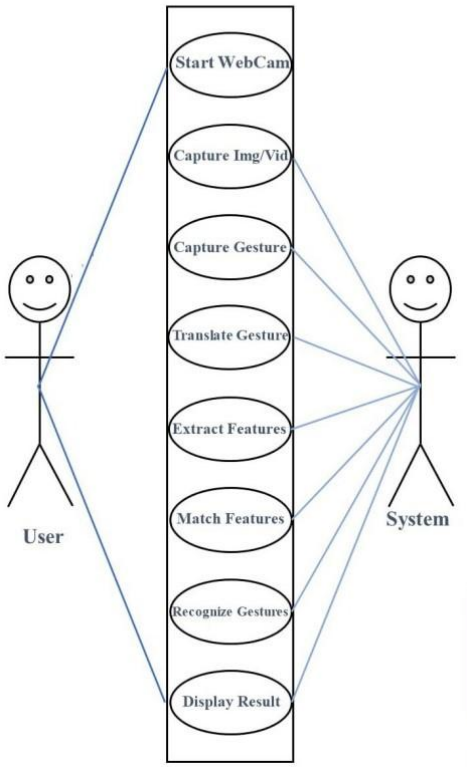


Fig 4.2.1 Use Case Diagram

4.3 Data Flow diagram

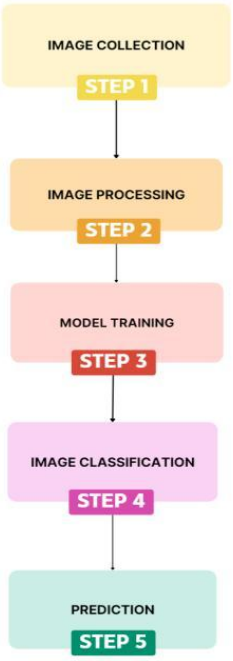


Fig 4.3.1 Data Flow Diagram

4.4 Sequence diagram

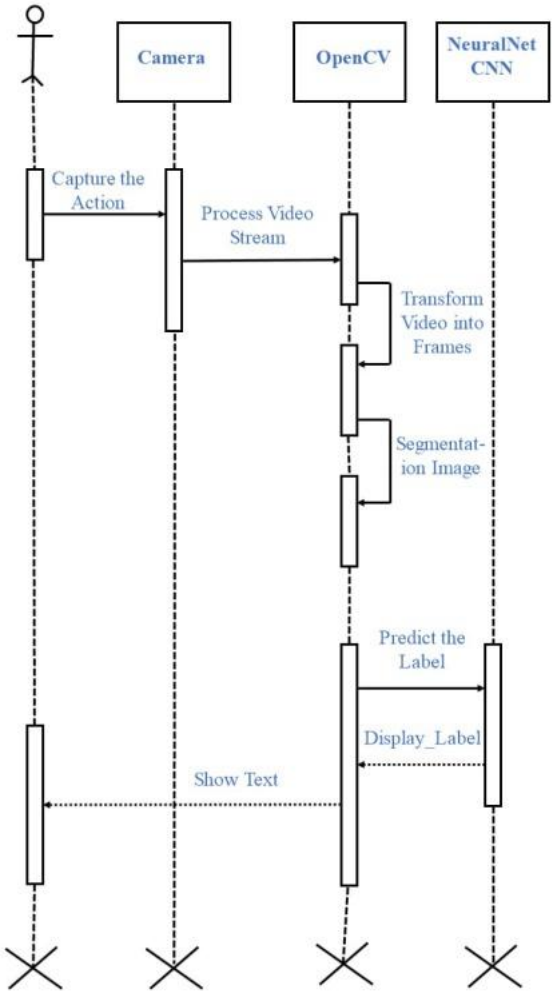


Fig 4.4.1 Sequence Diagram

Chapter 5

Implementation and Results

5.1 Implementation

Dataset Preparation:

Collect a dataset of sign language images with a diverse range of hand shapes and poses, representing different signs.Annotate each image with the corresponding sign label.

Split the dataset into training and testing sets.Preprocess the images by resizing them to a consistent size, such as 64x64 pixels, and normalize pixel values to a suitable range, such as [0, 1].Optionally, apply data augmentation techniques like rotation, translation, or flipping to increase the dataset's size and diversity.

Set up the Development Environment:

Install the required libraries: TensorFlow, OpenCV, Mediapipe, Matplotlib, and CVZone. Ensure that you use compatible versions.Import the necessary modules and dependencies in your Python environment.

Data Loading and Preprocessing:

Load the preprocessed dataset into memory, including the sign images and their corresponding labels.Split the dataset into training and validation subsets for model evaluation during training. Convert the labels into a numerical format suitable for training a CNN model

Build the CNN Model Architecture:

Design the architecture of your CNN model with 28 layers using TensorFlow.

Specify the input shape of the images (e.g., (64, 64, 3)) and consider using a convolutional layer with a suitable number of filters and kernel size.Stack multiple convolutional layers, followed by activation functions like ReLU.Utilize pooling layers (e.g., MaxPooling2D) to downsample the feature maps and reduce the model's spatial dimensions.Add additional convolutional and pooling layers as needed to increase model depth.Flatten the output feature maps to a 1D vector and connect them to one or more fully connected layers.

Consider using techniques like dropout or batch normalization to improve the model's generalization capabilities.Specify the output layer with the appropriate number of

neurons based on the sign classes and use an activation function suitable for multi-class classification (e.g., softmax).Train the CNN model using the training dataset. Use TensorFlow to feed the images into the model and adjust the weights based on predicted and actual values. Set the appropriate hyperparameters, including the number of epochs, batch size, and learning rate.Monitor the training process by evaluating the model's performance on the validation set. Adjust the hyperparameters or model architecture as needed to improve performance.

Evaluate the CNN Model:

Once the model is trained, evaluate its performance using the testing dataset that was set aside.Calculate metrics such as accuracy, precision, recall, or F1 score to assess the model's effectiveness.

Real-time Sign Language Recognition:

Use OpenCV and Mediapipe to capture live video frames from a camera or process pre-recorded videos.Apply hand detection and tracking techniques using OpenCV and Mediapipe to extract hand regions from the video frames.Preprocess the extracted hand regions in the same way as the training images (resize and normalize).Feed the preprocessed hand regions into the trained CNN model to recognize the sign language gesture.

```python
import cv2
from cvzone.HandTrackingModule import HandDetector
import numpy as np
import math
import time

cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=1)

offset = 20
imgSize = 300

folder = "D:\\DIP real time\\Data\\Saranghee"
counter = 0

while True:
    success, img = cap.read()
    hands, img = detector.findHands(img)

    if hands:
        hand = hands[0]
        x, y, w, h = hand['bbox']

        imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) * 255
        imgCrop = img[y - offset: y + h + offset, x - offset: x + w + offset]

        if imgCrop.shape[0] != 0 and imgCrop.shape[1] != 0:  # Check if imgCrop is not empty
            imgCropShape = imgCrop.shape
            aspectRatio = h / w

            if aspectRatio > 1:
                k = imgSize / h
                wCal = math.ceil(k * w)
                imgResize = cv2.resize(imgCrop, (wCal, imgSize))
                imgResizeShape = imgResize.shape
                wGap = math.ceil((300 - wCal) / 2)
                imgWhite[:, wGap: wCal + wGap] = imgResize
            else:
                k = imgSize / w
                hCal = math.ceil(k * h)
                imgResize = cv2.resize(imgCrop, (imgSize, hCal))
                imgResizeShape = imgResize.shape
                hGap = math.ceil((300 - hCal) / 2)
                imgWhite[hGap: hCal + hGap, :] = imgResize

            cv2.imshow("ImageCrop", imgCrop)
            cv2.imshow("ImageWhite", imgWhite)

    cv2.imshow("Image", img)
    key = cv2.waitKey(1)

    if key == ord("s"):
        counter += 1
        cv2.imwrite(f'{folder}/Image_{time.time()}.jpg', imgWhite)
        print(counter)
```

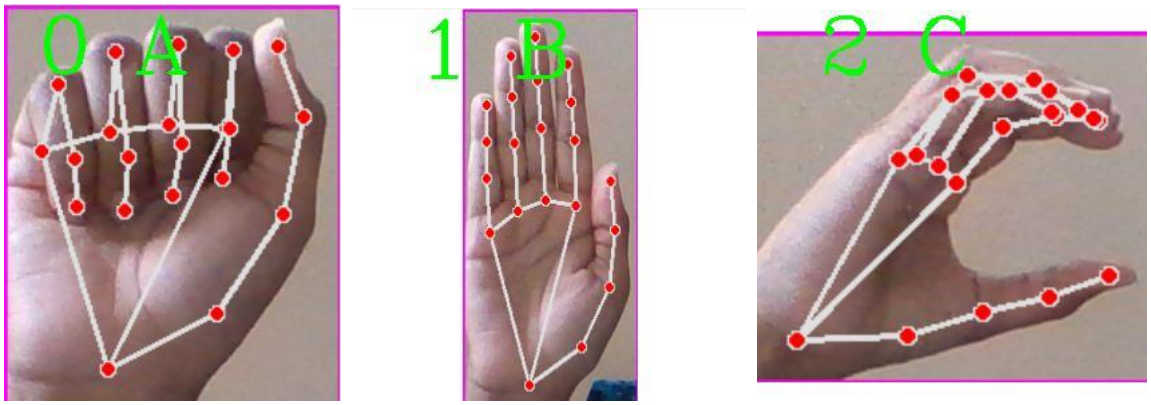
Fig 5.1.1 Data Collection & Data Processing

```python
import cv2
from cvzone.HandTrackingModule import HandDetector
from cvzone.ClassificationModule import Classifier
import numpy as np
import math
import time

cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=1)
classifier=Classifier("D:\\DIP real time\\Model\\keras_model.h5","D:\\DIP real time\\Model\\labels.txt")

offset = 20
imgSize = 300

folder = "D:\\DIP real time\\Data\\I"
counter = 0

labels = ["A","B","C","D","E","F","G","H","I","J","K","L","M","N","O"]

while True:
    success, img = cap.read()
    imgOutput = img.copy()
    hands, img = detector.findHands(img)

    if hands:
        hand = hands[0]
        x, y, w, h = hand['bbox']

        imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) * 255
        imgCrop = img[y - offset: y + h + offset, x - offset: x + w + offset]

        if imgCrop.shape[0] != 0 and imgCrop.shape[1] != 0:  # Check if imgCrop is not empty
            imgCropShape = imgCrop.shape
            aspectRatio = h / w

            if aspectRatio > 1:
                k = imgSize / h
                wCal = math.ceil(k * w)
                imgResize = cv2.resize(imgCrop, (wCal, imgSize))
                imgResizeShape = imgResize.shape
                wGap = math.ceil((300 - wCal) / 2)
                imgWhite[:, wGap: wCal + wGap] = imgResize
                prediction,index = classifier.getPrediction(imgWhite)
                print(prediction, index)


            else:
                k = imgSize / w
                hCal = math.ceil(k * h)
                imgResize = cv2.resize(imgCrop, (imgSize, hCal))
                imgResizeShape = imgResize.shape
                hGap = math.ceil((300 - hCal) / 2)
                imgWhite[hGap: hCal + hGap, :] = imgResize
                prediction,index = classifier.getPrediction(imgWhite)



            cv2.imshow("ImageCrop", imgCrop)
            cv2.imshow("ImageWhite", imgWhite)

    cv2.imshow("Image", imgOutput)
    cv2.waitKey(1)
```

Fig 5.1.2 Data Analysis & Data Prediction

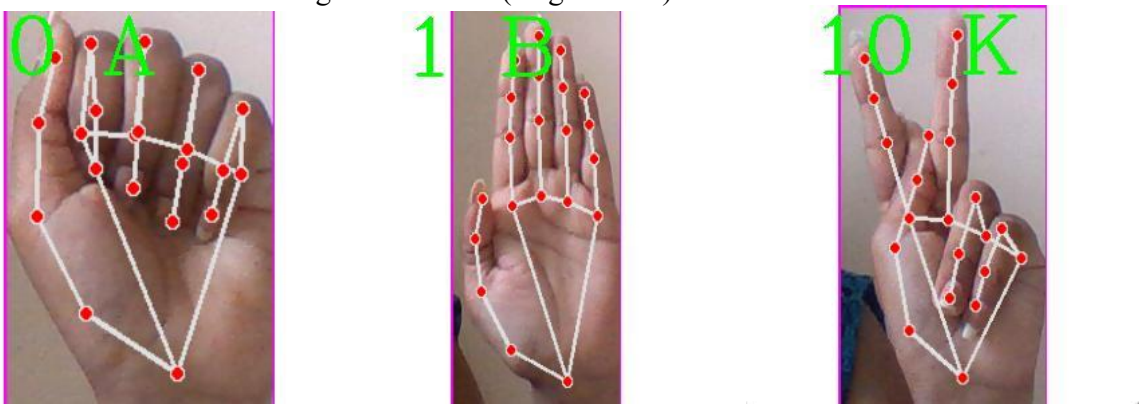
5.2 Result

In this project, With the help of American Sign Language (ASL) hand motions, we attempted to create a model in this research that could predict the alphabet from A to Z. We employed a variety of libraries and frameworks that are frequently used for computer vision and deep learning tasks to accomplish this Convolutional Neural Network (CNN) is a type of neural network that can process images and extract features. TensorFlow is an open-source platform for developing and training machine learning models. We used a sizable dataset of ASL hand gesture photographs representing the letters A–Z to train our model. Over 2000 photos make up the training dataset that will be used to train and test the sign language recognition model. The performance of our model was then assessed using fresh hand gesture photos (As seen in Fig 5.2.1 and 5.2.2) . The model was capable of identifying the alphabets on both the right and left hands.



(a) Alphabet A (b) Alphabet B (c) Alphabet C

Fig 5.2.1 Result (Right hand)



(a) Alphabet A (b) Alphabet B (c) Alphabet K

Fig 5.2.2 Result (Left hand)

The accuracy of the model was high for most of the alphabets, except for some of them that had similar shapes or orientations. For example, the letters G and H (Fig 5.2.3 and 5.2.4) and M and N (Fig 5.2.5 and 5.2.6) were not predicted accurately by the model, as they were confused with each other or with other letters. By gathering more photographs of hand motions in various lighting, background, and angle configurations, we were able to contribute more data for these specific alphabets, which improved the model's performance. We then retrained our model using the augmented dataset and observed an improvement in the accuracy of the predictions.

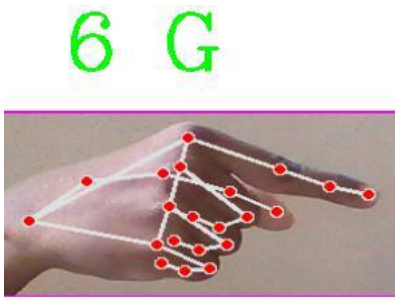


Fig 5.2.3 Alphabet G

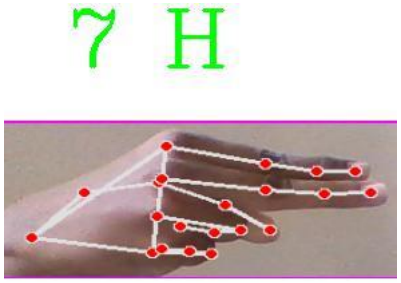


Fig 5.2.4 Alphabet H

Fig 5.2.5 Alphabet M

Fig 5.2.6 Alphabet N

Chapter 6
6.1 Hardware and Software requirements

Hardware Requirements	● A computer with a multi-core CPU ● High-end graphics card (GPU) ● Sufficient RAM would be required to train deep learning models on large datasets
Software Requirements	● Windows 32/64- bit operating System
Platform	● Windows 32/64-bit operating System
Programming Language/Tools	● Python ● TensorFlow ● Keras ● OpenCV ● Jupyter notebook

Conclusion

Sign language recognition utilizing CNN and related libraries offers a complete solution for understanding and deciphering sign language motions. These libraries make it possible to correctly and instantly recognize hand movements, enabling interactive apps and improving accessibility for sign language users. The creation of a more complicated model and the incorporation of natural language processing techniques are required for the expansion of the sign language recognition system to recognise whole sentences rather than individual gestures in future work. Additionally, adding speech synthesis and recognition skills would make it easier to translate sign language movements into spoken language, enhancing communication for people with hearing loss.

References

[1] P. Das, T. Ahmed and M. F. Ali, "Static Hand Gesture Recognition for American Sign Language using Deep Convolutional Neural Network," 2020 IEEE Region 10 Symposium (TENSYMP), Dhaka, Bangladesh, 2020, pp. 1762-1765, doi: 10.1109/TENSYMP50017.2020.9230772.

[2] M. M. Islam, S. Siddiqua and J. Afnan, "Real time Hand Gesture Recognition using different algorithms based on American Sign Language," 2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR), Dhaka, Bangladesh, 2017, pp. 1-6, doi: 10.1109/ICIVPR.2017.7890854.

[3] M. A. Rahim, J. Shin and M. R. Islam, "Dynamic Hand Gesture Based Sign Word Recognition Using Convolutional Neural Network with Feature Fusion," 2019 IEEE 2nd International Conference on Knowledge Innovation and Invention (ICKII), Seoul, Korea (South), 2019, pp. 221-224, doi: 10.1109/ICKII46306.2019.9042600.

[4] K. Bantupalli and Y. Xie, "American Sign Language Recognition using Deep Learning and Computer Vision," 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 2018, pp. 4896-4899, doi: 10.1109/BigData.2018.8622141.

[5] N. K. Bhagat, Y. Vishnusai and G. N. Rathna, "Indian Sign Language Gesture Recognition using Image Processing and Deep Learning," 2019 Digital Image Computing: Techniques and Applications (DICTA), Perth, WA, Australia, 2019, pp. 1-8, doi: 10.1109/DICTA47822.2019.8945850.

[6] C. Suardi, A. N. Handayani, R. A. Asmara, A. P. Wibawa, L. N. Hayati and H. Azis, "Design of Sign Language Recognition Using E-CNN," 2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT), Surabaya, Indonesia, 2021, pp. 166-170, doi: 10.1109/EIConCIT50028.2021.9431877.

[7] J. Krejsa and S. Vechet, "Czech Sign Language Single Hand Alphabet Letters Classification," 2020 19th International Conference on Mechatronics - Mechatronika (ME), Prague, Czech Republic, 2020, pp. 1-5, doi: 10.1109/ME49197.2020.9286667.

[8] A. Deshpande, A. Shriwas, V. Deshmukh and S. Kale, "Sign Language Recognition System using CNN," 2023 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE), Bengaluru, India, 2023, pp. 906-911, doi: 10.1109/IITCEE57236.2023.10091051.

[9] M. Safeel, T. Sukumar, K. S. Shashank, M. D. Arman, R. Shashidhar, and S. B. Puneeth, "Sign Language Recognition Techniques- A Review," *2020 IEEE Int. Conf. Innov. Technol. INOCON 2020*, pp. 1–9, 2020, doi: 10.1109/INOCON50539.2020.9298376.

[10] J. Wu, L. Sun, and R. Jafari, "A Wearable System for Recognizing American Sign Language in RealTime Using IMU and Surface EMG Sensors," *IEEE J. Biomed. Heal. Informatics*, vol. 20, no. 5, pp. 1281–1290, 2016, doi: 10.1109/JBHI.2016.2598302.

[11] K. L. Bouman, G. Abdollahian, M. Boutin, and E. J. Delp, "A low complexity sign detection and text localization method for mobile applications," *IEEE Trans. Multimed.*, vol. 13, no. 5, pp. 922–934, 2011, doi: 10.1109/TMM.2011.2154317.

[12] S. Hayani, M. Benaddy, O. El Meslouhi, and M. Kardouchi, "Arab Sign language Recognition with Convolutional Neural Networks," *Proc. 2019 Int. Conf. Comput. Sci. Renew. Energies, ICCSRE 2019*, pp. 2019– 2022, 2019, doi: 10.1109/ICCSRE.2019.8807586

[13] W. Aly, S. Aly, and S. Almotairi, "Userindependent american sign language alphabet recognition based on depth image and PCANet features," *IEEE Access*, vol. 7, pp. 123138–123150,2019,doi:10.1109/ACCESS.2019.2938829.

[14] Hao Zhou; Wengang Zhou; Yun Zhou; and Houqiang Li; Fellow "Spatial-Temporal Multi-Cue Network for Sign Language Recognition and Translation" DOI 10.1109/TMM.2021.3059098, IEEE 2022.